合成可能なタブローによる仕様の差分的無矛盾性判定について (プログラム変換と記号・数式処理)

AUTHOR(S):

友石, 正彦; 米崎, 直樹

CITATION:

友石, 正彦 ...[et al]. 合成可能なタブローによる仕様の差分的無矛盾性判定について (プログラム変換と記号・数式処理). 数理解析研究所講究録 2000, 1125: 142-149

ISSUE DATE:

2000-01

URL:

http://hdl.handle.net/2433/63575

数理解析研究所講究録
1125巻 2000年 142-149

合成可能なタブローによる仕様の差分的無矛盾性判定について

友石 正彦† 米崎 直樹†
Masahiko TOMOISHI Naoki YONEZAKI
†東京工業大学 大学院情報理工学研究科
Department of Computer Science, Tokyo Institute of Technology

これまでの時相論理による無矛盾性の判定方法では、仕様の部分についての検証結果を、全体仕様の検証に用いることができなかった。そこで本研究では、新たな構造を持ったタブローグラフとそれを用いる無矛盾性判定手続きを提案し、部分仕様について検証済みの情報を全体仕様の検証において利用できるタブロー法を提案する。特にこのタブローでは強連結成分をループの組み合わせとして表現し、それによる新しいイベンチャリティチェックの方法を導入する。本方法は、仕様が漸進的に記述されるときに部分仕様が追加される度に全体の検証を行う場合や、すでに検証済の部品の仕様を組み合わせて全体の仕様を構成する場合の検証において特に有効である。

1 はじめに

Until演算子を持つ時相論理のような順序を記述可能な時相論理を用いてリアクティブシステムの仕様を記述する研究が行われている[1]。リアクティブシステムは、エレベータシステムや、オペレーティングシステムなどのようにシステムが環境とやり取りを行う。多くのやり取りはあらかじめ決められたタイミングを満たす必要があるため、順序を記述可能な時間論理はそれらの時間的制約を記述するのに適している。また、時相論理という形式言語を用いて仕様記述を行うことで、厳密な記述が可能となるだけではなく、「仕様の段階での性質検証」が可能となるため、ソフトウェアプロセス全体の効率化を期待できる。

我々はさらに、仕様記述の過程において全体仕様が段階的に記述されていくものであることに注目した。仕様の性質検証を仕様が書き足されるたびに行うことができれば、仕様すべてを書き終わってから検証を行う場合に比べ、より早く問題部分の修正に書かれるだけでなく、修正すべき部分の特定もしやすい。しかし一般に、検証のコストは高く、繰り返し検証を行うのは現実的でない。そこで、我々は、以前の検証（タブロー法）で生成・使用された情報（タブロー）を使って検証する方法について研究を行ってきた（[4][5][6]）。

タブロー法（[2][3]など）は、よく知られた検証方法であり、特に時相論理で記述された仕様の性質検証の方法としては多くの研究があり、計算機上への実装も行われている。一般的に、タブローは時相論理で記述された仕様のモデルをパス（たどり方）とするグラフであり、そのグラフではノードが時間、枝が時間の遷移に対応する。ただし、すべてのパスがモデルに対応するわけではなく、タブローグラフ構成後にイベンチュアリティと呼ばれる時間的制約を検査することでモデルとなるパスがであるかどうか、つまり、与式にモデルがあるかどうかが判定される。

[4][5]ではタブロー法を改良し、仕様の差分と前回の仕様の検証における生成物であるタブローから新しい仕様に対するタブローを作る方

法を提案した。この方法では、一度調べた情報をタブローに反映しておくことで、同じことを2度調べないよう手続きが作られ、タブローを再利用する。しかし、この方法では、すでに部分に分けて記述された仕様について、その各部分の検証におけるタブローを用いて全体仕様を検証するためのタブローを構成することはできなかった。

そこで [6] では、部分仕様の検証において構成されたタブローから、部分仕様を合成して得られる全体仕様を検証するためのタブローを合成する方法について提案を行った。本研究では [6] での問題を修正したタブロー定義、手続きを提案する。

技術的には、まず、すでに生成されたタブローを合成することで新たなタブローを得るときに一度計算された情報を再計算しないで済むように、式の持つ命題への真偽の割り当ての情報だけではなく、時間制約の情報も反映した構造をタブローに持たせるようにした。具体的には、タブローは単にノードの集合とし、ノードは、リテラルだけからなるような基本となるノードを除いては、小さなタブロー2つとその間の到達関係から構成されるものとし、2つの小さなタブローと対応する部分モデルの情報とともに、その間の到達可能関係の情報も保持させる。このため、ノードの構造はこれまでのタブロー法におけるそれに比べ、極めて複雑なものとなる。

このタブローの構造は、一般的に知られたものや [4] などで構成したものと大きく異なり、無限語を受理するオートマトンとの対応 ([9][10]) がない。一般的なタブロー法では、オートマトンにおける受理条件に対応して、タブローグラフ構成後にイベンチャリティのチェック[1]を行なう。そのためノードは必ず $\langle A\rangle B$ という構造の式やその部分式 B がそのノードにあるという情報を保持する。つまり、$\langle A\rangle B$ の持つ情報の一部はタブローの構造に反映されるのではなく、ノードにラベルづけられた式に保持される。これに対し、本研究でのタブローでは、イベンチャリティによる時間の到達関係はノード内に保持される。また、[4] などでのタブローでは、追加仕様に対して前段階のタブローを展開し全体仕様のタブローを構成するために、タブローを動的な展開しなおすための情報として再展開される可能性のある部分に必要な式をノードに覚えておく必要があった。本研究の方法では時間の順序の再展開は行なわないため、このような情報は必要としない。

タブローの構成については、ノード内にある式に注目して動的に時間の順序を展開するのではなく、すでに順序の情報を反映した2つのタブロー（の要素であるノード）から1つのタブローを合成する。つまり、時間の順序のについてはすでに計算されたものの合成だけを行なう。これはこれまでのタブロー法において、同一の部分式の展開をタブロー構成の様々な場所で繰り返し行っているのに対し、ある部分式が持つ時間の順序の情報を先に計算し、それを全体に掛け合わせる。

また、最近のタブローに関する実装以外の研究として、ループの構成とイベンチャリティのチェックを効率化しようとするものがある [7], [8]。結果的に本研究の方法もループ構成とイベンチャリティのチェックに関してこれらと異なる技術を提供する。ループ構成は、これまでの手続きにおいて、手続きを停止させるためにすでにすでに作ったノードは再び作らずに戻り枝を作る方法であるが、これにより、新しくノードを構成する度にそれまでに作った全ノードとの比較が発生する。本手続きにおいてはノードが、意味的に未来から現在に向かって構成されるため他のノードとの比較の必要はなく、また、ループとなる可能性のある（部分）タブローは最初からループするものとして構成される。

イベンチャリティのチェックについては、イベンチャリティにあたる到達関係を保持するノードが、関係を保持したまま他のノードと合成できるかに対応し、関係を満たせないようなときには、合成手続きは空のタブローを返す。

[1] すべてのノード内にある $\diamond f$ に対して、到達可能なノード内にある f を見つける手続き

2 時間論理

本論文で使用する線形命題時間論理を以下のように定義する。

命題変数は式とする。f, g が式のとき、$\neg f$, $f \wedge g$, $[f]g$ は式とする。

$f \vee g$ は $\neg(\neg f \wedge \neg g)$ の省略形とし、$f \rightarrow g$ は $\neg f \vee g$ の、$\langle f\rangle g$ は $\neg[f]\neg g$ の省略形とする。

$\mathbb{N}$, m をそれぞれ自然数の集合、命題変数から $\mathbb{N}$ の部分集合への関数とするとき、$\langle \mathbb{N}, \leq, \mathsf{m}\rangle$ はモデルである。式 f について、モデル $M = \langle \mathbb{N}, \leq, \mathsf{m}\rangle$, $i \in \mathbb{N}$ における評価を ${}^{M}_{i}\models f$ と表し、以下のように定義する:

$$
\begin{array}{lcll}
{}^{M}_{i}\models p & \Leftrightarrow & i \in \mathsf{m}(p) & \cdots \quad p \text{は命題変数} \\
{}^{M}_{i}\models f \wedge g & \Leftrightarrow & {}^{M}_{i}\models f \text{ かつ } {}^{M}_{i}\models g & \\
{}^{M}_{i}\models \neg f & \Leftrightarrow & {}^{M}_{i}\not\models f & \\
{}^{M}_{i}\models [f]g & \Leftrightarrow & \left\{ \begin{array}{l} \forall j\,(i \leq j)\, {}^{M}_{j}\models g \text{ または、} \\ \exists k\,\{\, {}^{M}_{k}\models f \text{ かつ、} \\ \qquad \forall j\,(i \leq j < k)\, {}^{M}_{j}\models g\,\} \end{array} \right. &
\end{array}
$$

$[B]A$ はいわゆる弱い Until 演算子による A *Until* B と同じものである。

式が、任意のモデルの任意の状態で真であるときに恒真、あるモデルのある状態で真であるときに充足可能、任意のモデルの任意の状態で偽であるときに充足不能という。

3 タブロー法

Definition 3.1 (タブロー・ノード・ラベル) タブローはノードの集合とする。ノードは、リテラルの集合、または、二つのタブローとラベルからなる 3 つ組とする。ラベルは $\{\diamond, \circ\}$ の ($\emptyset$ を含めた) 部分集合とする。

以後、二つのタブロー $\mathsf{T}, \bar{\mathsf{T}}$ とラベル t からなるノードを $\langle \mathsf{T}, \bar{\mathsf{T}}\rangle^{t}$ と表し、そのときラベル t については $\{\}$ を省略して要素の列で表記する。

直観的には、$\langle \mathsf{T}, \bar{\mathsf{T}}\rangle^{t}$ は、タブロー $\bar{\mathsf{T}}$ に対応可能である部分モデルを何回か繰り返した後に、T に対応可能であるモデルに到達するようなすべての可能モデルの集合を表している。t は二つの部分モデルの間の到達に関する条件であり、$\diamond$ が必ず T に対応する部分モデルに到達する必要があること、$\circ$ が必ず $\bar{\mathsf{T}}$ に対応する部分モデルから始まる必要があることを表す。

例えば、$[a]b$ に対するタブロを構成すると

$$\{(\{\{a\}\}, \{\{\neg a, b\}\})\}$$

となり、$\langle a\rangle b$ に対するタブロは

$$\{(\{\{\neg a, b\}\}, \{\{\neg a, \neg b\}\})^{\diamond}\}$$

となるが (後に例で示す)、それぞれは、$\neg a, b$ である状態が続いて a である状態に到達するモデル、$\neg a, \neg b$ である状態が続いて $\neg a, b$ である状態に到達するモデルを表している。ラベルに $\diamond$ が含まれない $[a]b$ のタブローではずっと $\neg a, b$ である状態が続いても、つまり、ループであっても良いのに対して、$\langle a\rangle b$ のタブローでは必ず $\neg a, b$ である状態に到達する必要がある。また、$[a]b$, $\langle a\rangle b$ のタブローともにラベルに $\circ$ が含まれないので、それぞれ $a, \neg a, b$ である状態にいきなり到達する場合もタブロにより表現されたモデルである。

次に、ノードの除去条件と簡約の規則を定義する。本タブロー法では、ノードは 1 状態に対応するだけではなく、到達条件を満たす部分モデル (の集合) を連結したモデルにも対応する。そのため一般のタブロ構成規則と同じように相補リテラルを含むノードを除去するのに加えて、ラベルによって表された到達の条件を満たさないノードも除去する。また、ノードを取り除いた結果として、そのノードを含むタブロの対応する部分モデルがより簡単な構造のタブロに対応する場合には、そのより簡単なタブロで置き換える。

Definition 3.2 (ノードの除去条件) 除去の条件を満たすノードとは以下いずれかを満たすノードである。

1. 相補的リテラルを含む。

2. ノード $n = \langle \mathsf{T}, \bar{\mathsf{T}} \rangle^t$ が以下のいずれかを満たす。
(1) $\mathsf{T} = \bar{\mathsf{T}} = \emptyset$
(2) $\diamond \in t$ かつ $\mathsf{T} = \emptyset$
(3) $\circ \in t$ かつ $\bar{\mathsf{T}} = \emptyset$

1. はノードが一つの状態に対応する場合であり、部分式がリテラルまで分解された時点でノードが相補リテラルを含めば、ノードに対応する状態(部分モデル)は存在しない。2. はノードがタブローによって表された部分モデルの（到達可能条件を含めた）連結したモデルに対応する場合である。このとき、ノード $\langle \emptyset, \bar{\mathsf{T}} \rangle^\circ$ に対応するモデルはラベル $\diamond$ による「先のタブロー（が表す部分モデル）に必ず到達する」という条件を満たせない。また、ノード $\langle \mathsf{T}, \emptyset \rangle^\circ$ に対応するモデルはラベル $\circ$ による「前のタブロー（が表す部分モデル）から到達する」という条件を満たせない。よってこれらのノードは、ノード $\langle \emptyset, \emptyset \rangle$ とともに、全く対応するモデルを持たないノードなので、1. を満たすノードと同様にタブローグラフから取り除かれる。

Definition 3.3 (タブロの簡約規則) タブロの簡約規則とは以下とする。

$$\mathsf{U} \cup \{ \langle \mathsf{T}, \emptyset \rangle^t \} \quad \triangleright \quad \mathsf{U} \cup \mathsf{T} \qquad (\circ \notin t)$$

次にタブロの合成手続きにおいて $\diamond$ の条件を部分手続きに伝播させるために用いるラベルの合成を定義する。ここでの論理体系は線形離散時間を用いているため、ある未来に到達するためにその途中にある時間を必ず通る。そのため、ある時点からある未来に行く必要があるときには、その途中の時点でもその未来に行ける必要がある。タブロ合成手続きにおいては、未来にあたるモデルに対応するタブロとそこに到達する途中にあたるモデルに対応するタブロは独立に計算されるが、未来に行く必要があるという $\diamond$ の条件は途中の時点の計算にも反映されなければならない。

Definition 3.4 (ラベルの合成) t をラベルとするとき、d を以下のように定義する。

$$d(t) \equiv t \cap \{\diamond\}$$

t, u をラベルとするときラベル tu を以下のように定義する。

$$tu \equiv t \cup d(u)$$

ラベル tu は、t に加えて u から $\diamond$（による到達条件）を引き継ぐ。また、$\circ$ についてはそのような性質を持たない。

次に、2つのノードから、充足するモデル（順序）、到達条件を維持したまま合成してできるタブローを返す手続き $\times, \times_*^t, \times_\Box, \times_\diamond$ を定義する。

$\times$ は単純に二つのタブローの順序が維持されるようにノードを重ねるのに用いられる。

$\times_*$ は擬順序を表すタブローを重ねるのに用いられる。つまり、未来にすでにモデル（タブロー）がある場合に、その手前の部分モデル（タブロー）を合成するような、$\langle \mathsf{T}, \bar{\mathsf{T}} \times_*^{d(t)} \mathsf{U} \rangle^t$ といったノードにおいて用いられ、$\bar{\mathsf{T}}$ と U から作られるループ、つまり、互いの未来に互いの始点に戻ってもいいようなモデルの合成を実現するのに用いられる。また、$\times_*$ は第3の引数としてラベルをとる。これは先ほど述べた通り、合成するタブロに対応するモデルが未来に行く必要がある場合（たとえば $\diamond \in t$）には、そのすべての途中においてもその未来に行く必要があるためである。

$\times_\Box, \times_\diamond$ はそれぞれ $[\,], \langle\,\rangle$ に対するタブロー構成の際にのみ呼び出され、それぞれの意味を満たす順序に対応するモデルのみを合成するように用いられる。具体的には、$[A]B$ のタブローは、A になるまでは B であるようなモデルの集合でなければならないため、B のタブロが $\neg A$ のタブロと合成されることはない。また、$\langle A \rangle B$ のタブロでは、$\neg B$ であるの間と B になるそのときも $\neg A$ であるモデルを合成する必要がある。$\times_\Box, \times_\diamond$ は、これらの意味を反映して合成を実現する。

Definition 3.5 (合成手続き) 二つのノード、または、二つのタブローからひとつのタブローを返す手続き $\times$, $\times_\Box$, $\times_\diamond$ と、二つのノードと一つのラベル、または、二つのタブローと一つのラベルからひとつのタブローを返す手続き $\times_*$ を以下のように定義する。ただし、以下の手続き中、除去条件を満たすノードが現れたときには、そのノードを除去するものとし、タブローが簡約可能なときは簡約する。また、_はその位置に任意のものが来てよいことを表す *(don't care)*。

case 1. $\mathsf{n_t}$, $\mathsf{n_u}$ をそれぞれリテラルの集合 l_t, l_u とするとき

$$\mathsf{n_t} \times \mathsf{n_u} \equiv \{ l_t \cup l_u \}$$
$$\mathsf{n_t} \times^{\_}_* \mathsf{n_u} \equiv \{ l_t \cup l_u \}$$

case 2. $\mathsf{n_t} = \langle \mathsf{T}, \bar{\mathsf{T}} \rangle^t$、$\mathsf{n_u}$ がリテラルの集合 l_u のとき

$$\mathsf{n_t} \times \mathsf{n_u} \equiv \begin{cases} \{ \langle \{\mathsf{n_t}\}, \bar{\mathsf{T}} \times \{l_u\} \rangle^{\circ t} \} & \cdots \circ \in l_u \\ \{ \langle \{\mathsf{n_t}\}, \bar{\mathsf{T}} \times \{l_u\} \rangle^{\circ t} \} \cup \mathsf{T} \times \{l_u\} & \\ \cdots \text{それ以外} & \end{cases}$$

$\langle \mathsf{T}, \bar{\mathsf{T}} \rangle^t$ で表される順序とその時点での付値を表す l_u の合成を考えると、Tにいきなり到達する場合、Tと l_u が重なり、そうでない場合、まず $\bar{\mathsf{T}}$ と l_u が重なり、さらに $\mathsf{n_t} = \langle \mathsf{T}, \bar{\mathsf{T}} \rangle^t$ に対応するモデルが続く。またこのとき、$\circ$がラベル t に含まれていれば、いきなりTに到達するモデルを考える必要はないので候補からはずされる。

case 2.（続き）

$$\mathsf{n_t} \times_\Box \mathsf{n_u} \equiv \{ \langle \mathsf{T}, \bar{\mathsf{T}} \times^{d(t)}_* \{l_u\} \rangle^t \}$$
$$\mathsf{n_t} \times_\diamond \mathsf{n_u} \equiv \{ \langle \mathsf{T} \times \{l_u\}, \bar{\mathsf{T}} \times^{d(t)}_* \{l_u\} \rangle^t \}$$
$$\mathsf{n_t} \times^s_* \mathsf{n_u} \equiv \{ \langle \mathsf{T} \times^s_* \{l_u\}, \bar{\mathsf{T}} \times^{d(ts)}_* \{l_u\} \rangle^{ts} \}$$

$\times_\Box$ は $[A]B$ に対するタブローを構成する際に使われる。A までの間常に B である必要があるため $\times$ ではなく、$\times_*$ を用いて A になるまでの区間では l_u であるよう合成する。$\times_\diamond$ についても同様である。$\times_*$ は、例えば $\times_\Box$ からの呼び出しのように使われ、l_t の順序と関係なく全区間に l_u を合成する（用いられる計算がすべて $\times_*$ によるのですべての区間に l_u を合成する）。

case 3. $\mathsf{n_t}$, $\mathsf{n_u}$ をそれぞれ $\langle \mathsf{T}, \bar{\mathsf{T}} \rangle^t$, $\langle \mathsf{U}, \bar{\mathsf{U}} \rangle^u$ とするとき、まず $\mathsf{n_{uo}}, \ldots, \mathsf{n_{to}}$ を以下とする：

$$\mathsf{n_{uo}} \equiv \langle \langle \{\mathsf{n_t}\}, \bar{\mathsf{T}} \times \mathsf{U} \rangle^{\circ t}, \bar{\mathsf{T}} \times^{d(ut)}_* \bar{\mathsf{U}} \rangle^{ut},$$
$$\mathsf{n_{ut}} \equiv \langle \{ \langle \mathsf{T} \times \mathsf{U}, \bar{\mathsf{T}} \times^{d(t)}_* \mathsf{U} \rangle^t \}, \bar{\mathsf{T}} \times^{d(ut)}_* \bar{\mathsf{U}} \rangle^{ut},$$
$$\mathsf{n_{tu}} \equiv \langle \{ \langle \mathsf{T} \times \mathsf{U}, \mathsf{T} \times^{d(u)}_* \bar{\mathsf{U}} \rangle^u \}, \bar{\mathsf{T}} \times^{d(tu)}_* \bar{\mathsf{U}} \rangle^{tu},$$
$$\mathsf{n_{to}} \equiv \langle \langle \{\mathsf{n_u}\}, \mathsf{T} \times \bar{\mathsf{U}} \rangle^{\circ u}, \bar{\mathsf{T}} \times^{d(tu)}_* \bar{\mathsf{U}} \rangle^{tu}$$

このとき（図1）、

$$\mathsf{n_t} \times \mathsf{n_u} \equiv \{ \mathsf{n_{uo}}, \mathsf{n_{ut}}, \mathsf{n_{tu}}, \mathsf{n_{to}} \}$$

とする。

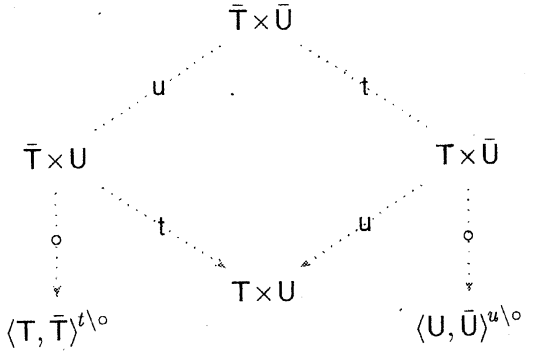

図 1: $\langle \mathsf{T}, \bar{\mathsf{T}} \rangle^t \times \langle \mathsf{U}, \bar{\mathsf{U}} \rangle^u$

$\times_\Box$, $\times_\diamond$ については、それぞれ [ ], ⟨ ⟩ の意味によって、$\bar{\mathsf{T}} \times^{d(u)}_* \mathsf{U}$ から $\bar{\mathsf{T}} \times^{d(u)}_* \bar{\mathsf{U}}$ へのループ（擬順序）がある（図2、図3）。

$$\mathsf{n_t} \times_\Box \mathsf{n_u} \equiv \{ \langle \mathsf{T} \cup \{\mathsf{n_{ut}}, \mathsf{n_{tu}}, \mathsf{n_{to}}\}, \langle \bar{\mathsf{T}} \times^{d(u)}_* \mathsf{U}, \bar{\mathsf{T}} \times^{d(u)}_* \bar{\mathsf{U}} \rangle^u \rangle \}$$

$$\mathsf{n_t} \times_\diamond \mathsf{n_u} \equiv \{ \langle \{\mathsf{n_{ut}}, \mathsf{n_{tu}}, \mathsf{n_{to}}\}, \langle \bar{\mathsf{T}} \times^{d(u)}_* \mathsf{U}, \bar{\mathsf{T}} \times^{d(u)}_* \bar{\mathsf{U}} \rangle^u \rangle^\diamond \}$$

case 3.（続き） $\mathsf{n}^*_1, \ldots, \mathsf{n}^*_4$ を以下とする：

$$\mathsf{n}^*_1 \equiv \langle \bar{\mathsf{T}} \times^s_* \mathsf{U}, \bar{\mathsf{T}} \times^{d(us)}_* \bar{\mathsf{U}} \rangle^{us},$$
$$\mathsf{n}^*_2 \equiv \langle \{ \langle \mathsf{T} \times^s_* \mathsf{U}, \bar{\mathsf{T}} \times^{d(ts)}_* \mathsf{U} \rangle^{ts} \}, \bar{\mathsf{T}} \times^{d(uts)}_* \bar{\mathsf{U}} \rangle^{uts},$$
$$\mathsf{n}^*_3 \equiv \langle \{ \langle \mathsf{T} \times^s_* \mathsf{U}, \mathsf{T} \times^{d(us)}_* \bar{\mathsf{U}} \rangle^{us} \}, \bar{\mathsf{T}} \times^{d(uts)}_* \bar{\mathsf{U}} \rangle^{tus},$$
$$\mathsf{n}^*_4 \equiv \langle \mathsf{T} \times^s_* \bar{\mathsf{U}}, \bar{\mathsf{T}} \times^{d(ts)}_* \bar{\mathsf{U}} \rangle^{ts}$$

このとき、

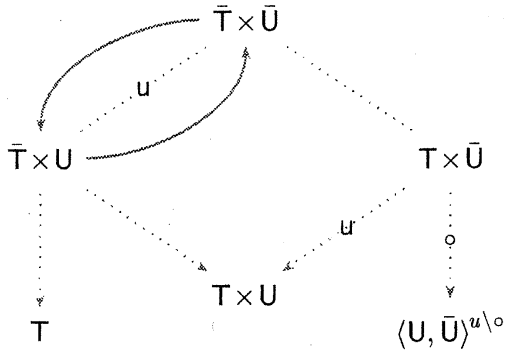

図 2: $\langle \mathsf{T}, \bar{\mathsf{T}} \rangle \times_{\Box} \langle \mathsf{U}, \bar{\mathsf{U}} \rangle^{u}$

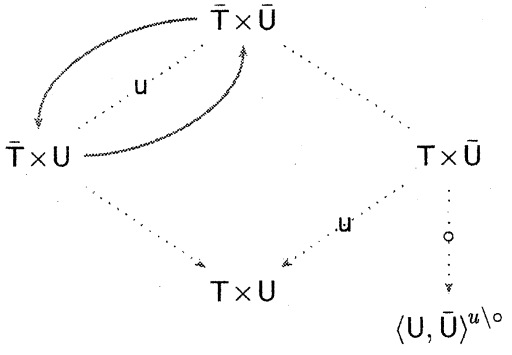

図 3: $\langle \mathsf{T}, \bar{\mathsf{T}} \rangle^{\Diamond} \times_{\Diamond} \langle \mathsf{U}, \bar{\mathsf{U}} \rangle^{u}$

$$\mathsf{n} \times_*^s \mathsf{n}'$$

$$\equiv \left\{ \begin{array}{l} \langle \{\mathsf{n}_2^*, \mathsf{n}_3^*\}, \{ \langle \{\mathsf{n}_4^*\}, \{\mathsf{n}_1^*\} \rangle^{\circ s}, \langle \{\mathsf{n}_1^*\}, \{\mathsf{n}_4^*\} \rangle^{\circ s} \} \rangle^{\circ s}, \\ \langle \{\mathsf{n}_2^*, \mathsf{n}_3^*\}, \{\mathsf{n}_1^*\} \rangle^{\circ st}, \\ \langle \{\mathsf{n}_2^*, \mathsf{n}_3^*\}, \{\mathsf{n}_4^*\} \rangle^{\circ su} \end{array} \right\}$$

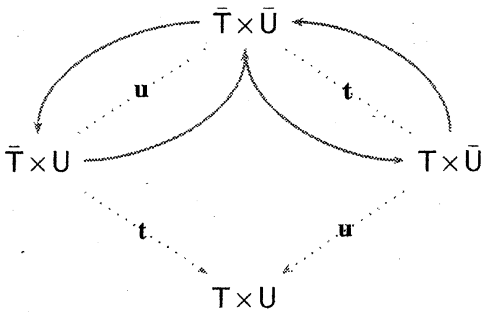

図 4: $\langle \mathsf{T}, \bar{\mathsf{T}} \rangle^{t} \times_* \langle \mathsf{U}, \bar{\mathsf{U}} \rangle^{u}$

$\times_*$ による合成では図 1 で n, n' に向かう到達可能関係の代わり、$\bar{\mathsf{T}} \times \bar{\mathsf{U}}$ へのループがある。

case T. T, T' がタブロであり、$\times_- \in \{\times, \times_*^t, \times_{\Box}, \times_{\Diamond}\}$ とするとき

$$\begin{array}{rcl} \mathsf{T} \times_- \emptyset & \equiv & \emptyset \times_- \mathsf{T} \equiv \emptyset \\ \mathsf{T} \times_- \mathsf{U} & \equiv & \bigcup_{\mathsf{n_t} \in \mathsf{T},\, \mathsf{n_u} \in \mathsf{U}} \mathsf{n_t} \times_- \mathsf{n_u} \end{array}$$

以上が重ねあわせの規則である。次にタブロー構成手続きを示す。

Definition 3.6 (タブロー構成手続き) 式からタブローを返す手続き T を以下のように定義する。

$$\begin{array}{lcll} T(l) & = & \{\{l\}\} & \cdots \quad l \text{ はリテラル} \\ T(\neg\neg A) & = & T(A) & \\ T(A \wedge B) & = & T(A) \times T(B) & \\ T(A \vee B) & = & T(A) \cup T(B) & \\ T([A]B) & = & \{\langle T(A), T(\neg A) \rangle^{\emptyset}\} \times_{\Box} T(B) & \\ T(\langle A \rangle B) & = & \{\langle T(B), T(\neg B) \rangle^{\Diamond}\} \times_{\Diamond} T(\neg A) & \end{array}$$

タブロー構成手続きにおいては、帰納的に部分式のタブローを計算するとともに、式の構造がタブローの構造に反映されるように式の構造に応じた合成規則が呼ばれる。

Definition 3.7 (タブロー法) 式fのタブロー$T(f)$が空でないとき、そのときに限り、「式fは充足可能である」と判定する。

Tの定義から$T(f \wedge g)$のタブローは$T(f) \times T(g)$として構成される。$T(f), T(g)$がすでに構成されていれば、$\times$の定義により、新たなタブロー構成手続きが呼びだされることはなく（$\times_\Box, \times_\diamond$も呼ばれない）$T(f) \times T(g)$が構成される。つまり、$T(f), T(g)$のタブローの生成において得られた情報はすべてタブローの構造に反映されており、それらについて再び調べることなく$T(f \wedge g)$の構成を行うことができる。

イベンチャリティチェックは、ノードの除去条件における$\langle \emptyset, T \rangle^\diamond$の除去と、$T(\langle A \rangle B)$によって作られるラベル$\diamond$を$B$の部分タブロー全体に伝播させることにより実現されている。

一般のタブロー法におけるタブロー構成におけるループ・イベンチャリティに注目すると、イベンチャリティを満たすことのできないようなループ（強連結成分）のグラフ生成時には、ループの各ノードにおいてイベンチャリティを満たすためのグラフ（ノード）の生成を行っている。しかしそれらはすべて取り除かれてループだけが残っているのである。つまり、各ノードにおいてイベンチャリティを満たすことができなかったことが既知であるにもかかわらず、それがタブロー中に情報として反映されておらず、ループ部分全体においてイベンチャリティを満たすことができなかったどうかもタブロー生成中に判断できないため、グラフ生成後にそれを調べる必要がでるのである。

本論文のタブロー法では、イベンチャリティを満たすためのノード（タブロー）を合成した時点で、そのノードが取り除かれるか（$\langle \emptyset, T \rangle^\diamond$）、または、そのノードに到達できなくなるか（$\diamond$の伝播）が判断可能であるため、そのイベンチャリティを必要とする部分グラフ全体をその場で取り除くことが可能なタブローグラフの構造・構成になっている。これによりグラフ生成後のイベンチャリティのチェック手続きはない。

4 例題

タブローを構成する例をあげる。ただし、以下では英数小文字は命題変数を表すものとし、アンダーラインはその要素からなるノードだけからなるタブローを表す。

$$
\begin{aligned}
T([a]b) &= \{\langle T(a), T(\neg a)\rangle\} \times_\Box T(b) \\
&= \{\langle \underline{a}, \underline{\neg a}\rangle\} \times_\Box \underline{b} \\
&= \langle \underline{a}, \underline{\neg a}\rangle \times_\Box \{b\} \\
&= \{\langle \underline{a}, \underline{\neg a} \times_* \underline{b}\rangle\} \\
&= \{\langle \underline{a}, \underline{\neg a}, b\rangle\}
\end{aligned}
$$

$$
\begin{aligned}
T(\langle a \rangle b) &= \{\langle T(b), T(\neg b)\rangle^\diamond\rangle\} \times_\diamond T(\neg a) \\
&= \{\langle \underline{\neg a, b}, \underline{\neg a, \neg b}\rangle^\diamond\}
\end{aligned}
$$

ラベル$\diamond$の条件により、以後の合成において$\langle a \rangle b$のタブロから$\{\neg a, b\}$が取り除かれれば、$\langle \underline{\neg a, b}, \underline{\neg a, \neg b}\rangle^\diamond$も取り除かれ、モデルがなくなる。

$$
\begin{aligned}
&T([a]\langle a \rangle b) \\
&\quad = \{\langle T(a), T(\neg a)\rangle\} \times_\Box T(\langle a \rangle b) \\
&\quad = \langle \underline{a}, \underline{\neg a}\rangle \times_\Box \langle \underline{\neg a, b}, \underline{\neg a, \neg b}\rangle^\diamond \\
&\quad \left(\begin{array}{l}
\mathsf{n_{ut}} = \langle \emptyset, \{\langle \underline{\neg a, b}, \underline{\neg a, \neg b}\rangle^\diamond\}\rangle \\
\mathsf{n_{tu}} = \langle \{\langle \emptyset, \emptyset\rangle^\diamond\}, \underline{\neg a, \neg b}\rangle^\diamond = \langle \emptyset, \underline{\neg a, \neg b}\rangle^\diamond \\
\mathsf{n_{to}} = \langle \langle \underline{\neg a, b}, \underline{\neg a, \neg b}\rangle^\diamond, \emptyset\rangle^{\{\diamond, \emptyset\}}, \underline{\neg a, \neg b}\rangle^\diamond \\
\quad = \langle \emptyset, \underline{\neg a, \neg b}\rangle^\diamond
\end{array} \right. \\
&\quad = \{\langle \{\{a\}, \langle \emptyset, \{\langle \underline{\neg a, b}, \underline{\neg a, \neg b}\rangle^\diamond\}\rangle\}, \\
&\qquad \{\langle \underline{\neg a, b}, \underline{\neg a, \neg b}\rangle^\diamond\}\rangle\}
\end{aligned}
$$

$$
\begin{aligned}
&T([a]\langle a \rangle \neg b) \\
&\quad = \{\langle \{\{a\}, \langle \emptyset, \{\langle \underline{\neg a, \neg b}, \underline{\neg a, b}\rangle^\diamond\}\rangle\}, \\
&\qquad \{\langle \underline{\neg a, \neg b}, \underline{\neg a, b}\rangle^\diamond\}\rangle\}
\end{aligned}
$$

$$
\begin{aligned}
&T([a]\langle a \rangle b \wedge [a]\langle a \rangle \neg b) \\
&\quad = \{\langle \underline{a}, \\
&\qquad \{\langle \emptyset, \\
&\qquad\quad \{ \\
&\qquad\qquad \langle \{\langle \underline{\neg a, b}, \emptyset\rangle^\diamond\}, \{\langle \underline{\neg a, \neg b}, \emptyset\rangle^\diamond\}\rangle^\diamond, \\
&\qquad\qquad \langle \{\langle \underline{\neg a, \neg b}, \emptyset\rangle^\diamond\}, \{\langle \underline{\neg a, b}, \emptyset\rangle^\diamond\}\rangle^\diamond
\end{aligned}
$$

$$
\begin{array}{l}
\quad\quad\quad\quad \}\\
\quad\quad\quad \rangle\}\\
\quad\quad \rangle\}\\
= \{\langle \underline{a},\\
\quad\quad \{\langle \emptyset,\\
\quad\quad\quad \{\langle \underline{\neg a, b}, \underline{\neg a, \neg b}\rangle^{\circ}, \langle \underline{\neg a, \neg b}, \underline{\neg a, b}\rangle^{\circ}\}\\
\quad\quad \rangle\}\ \rangle\}
\end{array}
$$

このタブローに、さらに$\diamond a$のタブロー$\{\langle a, \neg a\rangle^{\circ}\}$を重ねたものを考えると、後半部分が消え、$\{\langle \underline{a}, \emptyset\rangle\}$だけが残る。$[a]\langle a\rangle b \wedge [a]\langle a\rangle\neg b \wedge \diamond a$ のタブローは、これまでのタブロー法ではイベンチャリティをチェックをしなければ、ループ部分が表現するモデルを取り除けない典型的な場合であるが、本研究のタブロー法では合成時にこれを取り除くことに成功している。

5 まとめと課題

本研究では全く新しい構造を持つタブローと、それを用いるタブロー法を提案した。

この方法を用いることですでに検証済みの部分仕様がある場合に、それぞれの部分仕様検証時に生成されたタブローを用いて、部分仕様を合成した仕様の検証を、一度調べた情報を再び調べることなく行なうことができるようになった。

また、ここで提案したタブローの新しい構造は、イベンチャリティのチェック手続きを、これまでのすべてのイベンチャリティを表すの式$\langle f_i\rangle g_i$に対してそれぞれイベントg_iを見つけるという手続きから、$\langle \emptyset, T\rangle^{\diamond}$という構造のノードを取り除くという簡潔な手続きへ変更することを可能とした。

仕様が合成されていくのにともない、合成される各部分仕様に対して構成されたタブローを利用して全体仕様のタブローを構成することができるため、仕様とタブローによる検証を常に対にして考えることができ、仕様記述のプロセスは、検証結果を得るためのプロセスと同期する。これにより、開発プロセスにおける検証がより現実的なものとなったと考えられる。

課題としては、ノードの形が複雑なため手続きの正当性を証明するに至っていないことがあげられる。また、タブローを合成手続きにおいて再利用する場合、より小さなタブローが有効であるので、それを求める方法について検討したい。

参考文献

[1] E.Allen Emerson and Jai Srinivasan. Branching time temporal logic. *Lecture Note in Computer Science*, Vol. 354, , 1988.

[2] E.Allen Emerson and Joseph Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Science*, Vol. 30, pp. 1–24, 1985.

[3] Albert Zanardo. A complete deductive-system for since-until branching-time logic. *Journal of Philosophical Logic*, Vol. 20, pp. 131–148, 1991.

[4] 友石正彦, 米崎直樹. 動作仕様の差分的無矛盾判定. 第 10 回大会論文集, pp. 217 – 220. 日本ソフトウェア科学会, 1993.

[5] 友石正彦, 三木一秀, 米崎直樹. リアクティブシステム仕様の合成による無矛盾性証明法. 第 14 回大会論文集, pp. 345 – 348. 日本ソフトウェア科学会, 1997.

[6] 友石正彦, 米崎直樹. 合成可能な時間論理タブローの構成法. 電子情報通信学会技術研究報告, Vol. 97, No. 390, pp. 17–22, 1997.

[7] Stefan Schwendimann. A new one-pass tableau calculus for pltl. In *Tableau'98, LNCS*, Vol. 1397, pp. 277–291, 1998.

[8] M. Seyfried A. Heuerding and H. Zimmermann. Efficient loop-check for backward proof search in some non-classical propositional logics. In *Tableau'96, LNCS*, Vol. 1071, pp. 210–225, 1996.

[9] M.Vardi and P.Wolper. Automata theoretic techniques for modal logics of programs. In *Proc. 16th ann. ACM Symp. on Theory of Computing*, pp. 446–456, 1984. J. Comp. System Sci. 32 (1986) 183 – 221.

[10] E. Emerson and C. Jutla. The complexity of free automata and logics of programs. In *Symp. on Foundations of Computer Science*, Vol. 29, pp. 328 – 337. IEEE-CS, 1988.